Transactions of the SDPS:
Journal of Integrated Design and Process Science
18 (1), 2014, 21-38
DOI 10.3233/jid-2014-0008
http://www.sdpsnet.org

CASA - Context-Aware Service Access

Philipp Lehsten*, Sebastian Bader and Djamshid Tavangarian

Institute of Computer Science, University of Rostock, Rostock, Germany

Abstract With the widespread of mobile Internet-enabled devices we can consider nearly every user everywhere as a potential consumer of cloud services. Especially campus environments provide a multitude of heterogeneous scientific information and services to the users. Here different problems arise concerning the integration and dissemination of cloud services and resources. While location is usually considered as the core context for mobile service access, its value is very limited in evolving large-scale smart environments with a multitude of user roles, tasks, devices and services. This is even more challenging if multiple users work collaboratively in remote locations on the same project. Therefore, we propose with CASA, Context-Aware Service Access, a concept which can use different kinds of context as source and provides support for reactive and proactive actions based on the current situation. The evolving and changing aspects are handled with an extensive usage of crowdsourcing to encourage a decentralized and user-driven development. This shall lead to a novel open and extendable recommender system which also supports the adaption and integration of cloud-based services according to the situation. Additionally, as proof of concept, we present two use cases for smart campus and smart lecture rooms.

Keywords: Context-aware system, service integration, service dissemination, crowdsourcing

1. Introduction

The high dissemination of wireless networks and mobile Internet-enabled devices has enhanced the way of using the Internet. Mobile users still access the Internet to get news and access their mailbox, but as a matter of course they also look for situation dependent information in their current local area. Well-known examples are weather forecasts and price finders. But also the user role and the actual tasks define the scope of relevant services. Here the role of a lecturer in an upcoming lecture or the currently running simulations on a dedicated cluster provide relevant context information for the provision of cloud-based services like learning management systems or the cluster administration panel. While these examples contain mainly standard web sites, the upcoming Internet of Things (Atzori, Iera, & Morabito, 2010) introduces besides object-specific information also a multitude of services which extend the complexity of doing research in any field which uses hardware sensors or actors.

One of the main questions to handle this complexity is how to organize the services which already surround the user as well as those which are about to come. Here a concept is needed which is able to facilitate the discovery and access for the user as well as the service dissemination for the service provider.

*Corresponding author. Email: philipp.lehsten@uni-rostock.de Tel: (+49)381-498 7546.

To define the problem more precisely, the following five challenges (which are already there, and which are about to gain importance in the near future) are formulated:

C1. How can a mobile user employ and interact with ubiquitous services?

C2. How can a mobile user discover services provided for the current location, role and tasks?

C3. How can the discovered services be filtered automatically to identify the relevant ones?

C4. How can a service in a heterogeneous environment be used by a user who is not even aware of the available devices and does not want to be aware of it?

C5. How can a dynamic ecosystem of heterogeneous devices and services be managed and maintained if there is no central domain or infrastructure?

As approach to these challenges, the concept of a context-aware service access (CASA) is presented in this article. It handles different kinds of context to offer services according to the current situation in general and not only to the location. In our approach, the situation also includes information which can not be derived from the location alone, like the role of the user in a meeting or other complex information. Therefore different context sources from different domains must be requested. To connect this multitude of sources, a method is needed which allows the system to grow dynamically and to access context information of different kinds. This includes a simple integration of new sensors or other information sources, as well as new rules to interpret situations. Because of this, a core aspect of the system is the crowdsourcing of its parts, as this allows the community to extend the system with relevant sources and the knowledge to use them. Therefore, development interfaces for the user community are needed to add and to modify, and by this to evolve the different parts of the system (filters, locations, services and context sources).

In the next section some important observations are discussed. Additionally, the difference between the CASA approach and other published works is discussed. The CASA concept is fully introduced in section 3, while examples and use cases are presented in section 4. In section 5 the approach is discussed and possible next steps are identified.

2. Observations and related works

The CASA approach is driven by the challenges formulated above. Therefore, the observations which led to the approach are discussed first. In addition, related works illustrating alternative approaches are also discussed here.

2.1. Observations

The first challenge regards the accessibility and the interaction concepts for ubiquitous services (C1). These services are mostly hosted in different networks, therefore the interaction requires a device with at least one network interface. The cheap availability of wireless networks and the multitude of easy-to-use mobile Internet-enabled devices such as laptops, tablets, and smartphones lead to a web-based approach. Many mobile devices are additionally equipped with GPRS modules and GPS to provide location-aware Internet access and navigation especially in outdoor environments. As these networks can be considered today as nearly everywhere ubiquitous access is possible by these devices. While they can consume most web sites and additionally provide context like location there are also restrictions. As these aspects are already considered by developers, the consumable web-based services are multifaceted, heterogeneous and therefore we note that:

O1. Cloud services are ubiquitous and can be accessed using most mobile devices.

This leads to the question how to find the services which are relevant and usable to the user in his current situation. As the discovery of interesting services is essential for further steps, it represents the second challenge (C2). A user, standing in front of a university building, can be seen as a possible consumer to all services and information which are provided by the different departments and facilities. This is of course only a coarse filtering but it allows also to reduce the number of possible services. So we note:

O2. Location allows only a limited preselection of potentially important services and information.

Even if we assumed that there is a site which contains all provided services according to a given location, the user would have problems caused by the density of possible services. These could be timetables of lectures, opening hours of offices, the menu of the cafeteria, as well as role and task specific services like administration consoles of servers, collaborative writing services or reservation systems for (computational) resources. This leads to the question of how to filter the services for the user and identify the relevant ones: the third challenge (C3). Location is, as mentioned above, a possible filter, but the usability can be compared with a phone book. In environments with only a small number of services, e.g. less than 10, it is possible to get the relevant ones by location-based filtering, but in large-scale environments it is not sufficient, as a location, like a shopping mall or a university building, can easily provide thousands of services. Therefore the situation should not only be determined by the user's location, but also by identity, the time, and the activities he is involved in, which as a whole can be seen as context (Dey & Abowd, 2000). In large and heterogeneous environments with different services providing similar functionalities a cascade of context filters is needed to provide a list of relevant services, therefore we note:

O3. The usage of available services depends on the user's context. Therefore, the context as a whole should be used to identify the important ones.

Assumed that the user is able to discover and identify the relevant services on his own – how can these services be used? A web-based approach leads to a multitude of problems. Existing devices and web technologies reveal difficulties in the point of supported technologies like Flash or HTML5, but also in the point of adaption to the device constrains, like screen size or network bandwidth. Therefore, the usage of services is our fourth challenge (C4). Releasing the idealized campus environment with its high trained technicians, its centralized infrastructure and technically experienced users and moving to an environment considering different locations, like institutes and companies, we face buildings hosting different institutes, heterogeneous infrastructures, and a wider range of possible services and user devices. As this is the status quo, also the upcoming Internet of things, where every object can provide and offer its own functionalities as services, needs to be considered. Here it is essential for users to have homogeneous interfaces, which allow some kind of common standards in interfaces also above different service providers. Therefore, we note:

O4. Different services and information need to be combined into a homogeneous platform-independent representation to be accessible by different users.

Changing the point of view from the consumer-centric one to the provider-centric one, we need to ask how to offer services in a manner which allows them to be found by users who consider them as relevant? This dissemination of services and their maintenance is our fifth challenge (C5). To tackle this, we advocate an open system which is able to combine information of services from different providers. Community-driven projects like Wikipedia and OpenStreetMap have shown that reliable information, which has been collected and evaluated, can be provided by different users and can connect data of different domains. We also note that crowdsourcing may introduce new problems as the provided information is of heterogeneous quality and reliability. Also a user, who is able to add services, might not also have the knowledge to provide rules or create ontologies. Otherwise we think that if the system is open and accessible, there will be users who will have this knowledge. Additionally, we think that the distribution of these users across different domains can help to generate rules which are adaptable and ontologies which are consistent. The risk of intentionally misusage of such open systems is also there and needs to be addressed. Here we think of methods to sign rules and services to avoid vandalism and abuse.

O5. Crowdsourcing might be applicable to define the context, suitable filters and actions to be taken by the system.

Our approach is based on the usage of web-based interfaces to a system which filters the services by the user's context and the involvement of the crowd as distributed group of developers and maintainers.

The aspect of context recognition by sensors is abstracted as they only provide the raw data for the system. This keeps the system extensible and allows for instance different positioning methods in indoor environments. Here additional positioning hardware (Patterson et al., 2004) or more context is needed as filter. We prefer the latter but do not exclude specialized hardware as both methods can be used to estimate the user's location inside a building. As users are able to classify the situation in which they would like to access which service best, the system should provide interfaces which enable them to create filters and share them with the community. These interface can also be passively used by collecting the often used services. Here mechanisms like fuzzy inference systems might help to use this data. This might allow a fine-grained filtering where it is needed while keeping the information up-to-date.

As access method we advocate a web-based approach where interfaces to services are aggregated on a web site without the usage of technologies like Flash or other plugin-depending technologies. This is suitable for all mobile devices which are able to display a web site while only a browser is needed as software. Here the relevance can be communicated by the ordering or symbols. For services which do not provide own web sites, adapters are needed to translate their functionality into a (part of a) web site. These adapters as well as services, filters and context sources need to be developed and collected by a community. This decentralized and community-oriented approach will be able to maintain the system as it grows and changes with the devices and services it provides. Systems which evaluate the context on-demand to provide services to the user are only suitable in situations he is aware of a possible assistance. Therefore, we see the need for a proactive element in the system, as there are situations where the user needs to be notified of a context or situation where a service or information could influence his behavior.

As conclusion of our observations we advocate an open, decentralized system which is community-driven. It should allow the user to add services, situations, locations, filters, and context to provide the services the users treat as relevant. As representation we advocate for a web-based platform-independent approach which can be accessed by every Internet-enabled device with a browser.

2.2. Related work

When examining frameworks or middleware for context-aware usage of services, there is a multitude of research projects, which has been surveyed multiple times (Baldauf, Dustdar, & Rosenberg, 2007; Bellavista, Corradi, Fanelli, & Foschini, 2013; Kjær, 2007; Knappmeyer, Kiani, Reetz, Baker, & Tonjes, 2013). As this is a wide field, we will concentrate on the already identified challenges in this topic as well as on recently published works that share some facets with our approach.

In (Kiani, Moltchanov, Knappmeyer, & Baker, 2011) several different challenges for large-scale context-aware systems are discussed. The authors focus on the coordination mechanism, as a system which is not able to compete with the issues like scalability in its context model, or communication will fail. Additionally, it is stated that mobile devices are essential for context-aware applications as they are already part of the users life. Therefore, energy consumption of context-aware applications needs to be considered as well.

As conclusion they advocate a system which is scalable geographically, administratively, and functionally. This shall be achieved by a decoupling of the interacting components while keeping central components like a context broker as coordination instances. These requirements are also addressed by our system which additionally considers the maintenance and development as issues and therefore uses the crowd for system refinements.

An approach for a system which also focuses on the distribution and access of context-aware services is presented in (Chon & Cha, 2011). Here the smartphone is the main sensor of the system and therefore only the limited context, which can be provided by this system, is accessible. This allows a sensing of location, activity in the meaning of user motion, connectivity, environment with temperature, speed, and surrounding devices. The scalability is very good as every device is its own sensor but other context like user roles or user intentions can not be sensed and therefore the system has a constricted usability if there are multiple services provided by the same location.

The system presented in (Strobbe et al., 2011) has multiple ideas in common with our approach, especially the aggregation and abstraction of context by using ontologies, the rule-based reasoning for information validation, and deriving the information to high-level context. Also the idea to have a standardized exchange format to communicate with the sensors and then using an object-oriented format to communicate to upper context consumers is analogue. The difference here is that the system does only sense, evaluate and communicate the context, but there is no way mentioned to influence the way of this system from the user side. Additionally the system has no direct relations to the applications that use the context and therefore can not improve itself. The integration and access to the services are not covered as the system focuses on the context handling.

There are multiple approaches to classify human activities by mobile devices. Often the embedded accelerometer is used as it provides characteristic data for special activities like walking, biking, or standing (Brezmes, Gorricho, & Cotrina, 2009; Chon & Cha, 2011; Henpraserttae, Thiemjarus, & Marukatat, 2011). The combination of fuzzy inference systems and crowdsourcing can also be used to sharpen the results and to improve the performance and recognition of algorithms (Berchtold, Budde, Gordon, Schmidtke, & Beigl, 2010). From our perspective, these methods to classify activities only provide a limited description of the situation as they only use the sensors on the mobile device. We also intend to include software sensors in other domains. Although these recognized activities could be used as additional information for our system.

Finally, there are many different commercial recommender systems which already try to use the context they can gather to recognize the interests of the users. At the moment, Foursquare is a large system. It combines a location-based social network with a competition approach. The users are encouraged to visit user created locations to gain points. These points can be used to see which friend is the most active user. Additionally, the system awards virtual badges to the users which have met special conditions like 1000 check-ins. The recommending functionality is only location-based as Foursquare allows users to recommend places and activities there as lists which are attached to a location. As Foursquare is only location-based, the only contents which can be provided are locations, recommendations and comments concerning locations.

As far as we know, our approach of using crowdsourcing to collect context sources and rules is novel. The idea of combining this to an open system contains different risks, which were already mentioned, but it also may provide a way to integrate the different activity recognition systems into a system which could act as foundation for new applications.

In this paper, we are mainly concerned with the general design of such a system. In addition to general design choices, we are concentrating in particular on those which help a user-driven development of parts of the system. In the following section, we discuss the CASA approach on an abstract level and discuss two realised use cases afterwards.

3. CASA

Below, the CASA approach is described in some detail. As mentioned above, it is an approach to allow context-aware access to services. It features are based on the observations discussed in section 2:

- Open system able to combine information from different providers
- Community-driven to realize the maintenance and incremental development
- Evaluation of different kinds of context to allow a fine-grained filtering
- Web-based user interface to reach platform-independently every mobile internet-enabled device
- Decentralized gathering of context to provide scalability
- Proactive usage of context to allow also notifications in specified situations

Using the CASA approach, we intent to present an aggregation of currently important services and information to the user. In the examples, described below, we use websites as final representation to allow access to the system using various mobile devices. But the CASA approach itself is not limited to the generation of websites. Our system consists of the following parts detailed below:

- *Actions*. Context dependent actions (referred to as CActions) supporting the user in the current situation.
- *Context-Representation*. A representation of the current context, suitable to allow inference.
- *Context-Services*. Services interpreting the current context (referred to as CServices) to infer interesting and important CActions.
- *Aggregation*. After identifying the important CActions, the results need to be aggregated and to be presented to the user.

As mentioned above, the CASA system is designed as an open system. To meet this assumption, we neither require the CActions, nor CServices to be fixed. On the contrary, we assume them to be adapted dynamically by the user community. Therefore, we have to identify the important CActions and CServices depending on the user's current situation.

We use the term *context local* (CLocal) to refer to the CActions and CServices which are important with respect to the current location of the user.

Core of the architecture is the CASA node as context transformation and processing unit. As illustrated in figure 1, the node consists of a rules engine and a context storage as well as of different importer and aggregation modules for the usage of context and services. The importer modules are the connection to different services and context providers. Therefore, they translate their content and context to the context which is used by the CASA node. At this point, the raw context, e.g. sensor readings, which was collected by the importer modules, is transformed into low-level context, e.g. location updates. The collected data is transferred to the context storage. Here the information is evaluated using the CServices, which can be seen as the rules of the system. The available services and their encapsulated functionalities are translated into CActions. In accordance to the CServices this information and the services are offered to the importer modules of other CASA nodes or aggregation modules. This information forms now high-level context for the actual CASA node, because it is evaluated and connected with data from different sources. For other nodes this data is raw context which need to be transformed according to their own CServices via own importers.

The sensors and their APIs are proprietary and can usually not be changed by the users of the system. Context storage and the rules engine are specific to each node, so different nodes can use different systems for these modules. The knowledge and the usability of the system which is encapsulated in the importer and aggregation modules, as well as the CActions and CServices, are user-created and therefore they can

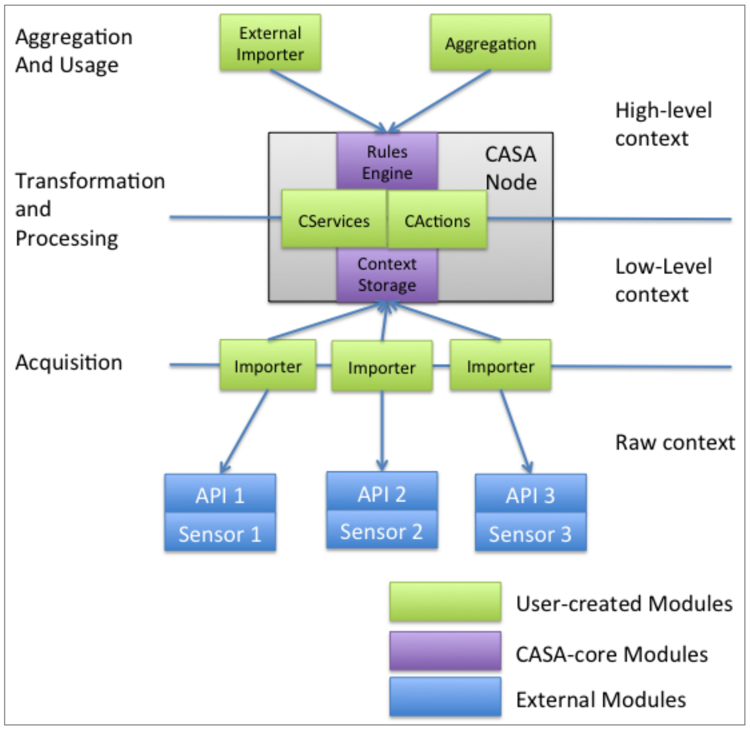


Fig. 1. Conceptual architecture of CASA nodes.

be interchanged as long as the used node can handle them. This is one of the core aspects as it allows a decentralized development and maintenance.

After describing the CASA-approach on an abstract level here, we discuss two implementations as proof of concept. Most universities use learning management systems (LMS) to digitally offer materials used in lectures. Some of these systems are also used to register students for courses or exams as well as to administrate timetables or rooms. Therefore, they reproduce workflows for different user groups. Our idea is to use a context-aware system to provide situation relevant services to the users of the LMS. For teachers this means it should integrate the services provided by devices in the lecture room (e.g. digital projector), whereas a student should be able to access a stream of the lecture or additional materials. Many of those systems are already web-based which reduced the implementation effort. As a second use case, we discuss the control of a smart meeting room offering different services to its users. A web-based interface allows to add new services into the room and to invoke existing ones.

Below, we first introduce CActions, providing the actual services to the user, and we discuss representation as well as manipulation of context. Then we discuss context services which map current context to suitable CActions and the aggregation of results. Finally, we show how to combine the different parts into one system addressing the challenges listed above.

3.1. CActions

As mentioned above, the CASA approach is meant to provide (proactive) assistance to its users. For this, it has to take certain actions depending on the current situation. CActions are those actions taken by the system. They basically can be everything which helps the user to achieve its goals. Potential types are:

- Notify the user by means of short messages, small popups or other means
- Allow direct interaction with local devices

- ○ Allow the invocation of services helpful in the current situation

CActions basically answer the questions '*What might the user want?*' and '*What should the user know or do?*' with respect to the current situation. As user-contributed part of the system, the set of CActions might change over time, but the available types are defined by the concrete implementation. The implementation discussed as our running example provides the CAction-types listed above, that is notifications, device interaction and service invocation. As discussed below, CActions are selected based on the current situation by context services. Those map the current context to a set of suitable CActions. We use $\mathscr{A}$ to refer to the set of all possible CActions. New CActions can be added to the system by integrating new services into it, or by implementing new device or service adapters.

3.2. Context

As mentioned above, CActions are provided to the user based on the current context. In the general CASA approach discussed here, we neither restrict the representation of the context, nor the inference procedure used to reason upon it, but leave it to the system implementation. But the chosen representation must be able to infer the set of useful CActions efficiently. The current context could, for example, be represented as a set of (object,property,values) triples as follows: $\{(\text{room}_1, \text{temp}, 30), (\text{room}_2, \text{temp}, 50), \ldots\}$. Below, we use $\mathscr{C}$ to refer to the set of all possible contexts.

To update the current context, we use *sensors*. Those incorporate updated information into the context store. As for the CActions, we assume sensors to be contributed by the user community. Therefore, we need to select the currently important sensors first. This can be done by annotating the sensors with locations they are responsible for. For example using an own server with *OpenStreetMap* users can attach objects to locations. This server can then be considered as sensor as it allows to query locations for objects and other geographical information. All local sensors can be queried to contribute to the users context.

A sensor s is a function on the context, mapping the current to a new state:

$$s : \mathscr{C} \to \mathscr{C}$$

A temperature sensor for room$_1$ would map the current context to a new one by updating the temperature triple for room$_1$.

Sensors are related to locations to define a set of relevant sensors. A sensor that provides the opening hours of the library would be connected with the location of the library. New sensors can be contributed to the system by implementing the sensor interface and attaching the sensor to a location. Afterwards, the sensor is found by the system and queried to update the internal database.

3.3. Context service

Context services interpret the current context and map it to important CActions. For example, there could be a service which results in a CAction showing available meals to all users near a given restaurant.

A *context service, referred to as CService,* maps the current context $c \in \mathscr{C}$ to a finite set of pairs (a, p) containing a CAction $a \in \mathscr{A}$ and a preference $p \in \mathbb{R}$. We use $\mathscr{S}_{\mathscr{C}}$ to denote the set of all CServices[1]:

$$\begin{aligned} \mathscr{S}_{\mathscr{C}} : \mathscr{C} &\to \mathscr{P}(\mathscr{A} \times \mathbb{R}) \\ c &\mapsto \{(a_1, p_1), (a_2, p_2), \ldots, (a_n, p_n)\} \end{aligned}$$

As above, the CASA architecture does not force a certain interface to be implemented, but of course, the CServices must be compatible with the chosen context representation.

[1]We use $\mathscr{P}(S)$ to denote the powerset of the given set S.

As for sensors and CActions, we assume context services to be user contributed. Every user can add new rules to the system by attaching rules including preference to a location. If the user is at the location, all registered context services are queried for their results. Please note that a context service being local does not imply that it actually produces a set of CActions, e.g. if the precondition of its rules is not satisfied, no CAction is returned. After collecting all resulting CActions preference pairs, they are aggregated to be presented to the user.

3.4. Aggregation

After querying all local context services, the resulting CActions need to be presented to the user. For this we assume a suitable representation $\mathscr{R}$ equipped with a composition operation $\circ : \mathscr{R} \times \mathscr{R} \to \mathscr{R}$ allowing to combine different parts of the representation. We furthermore assume a representation function $\mathrm{rep}_a : \mathbb{R} \to \mathscr{R}$ to be attached to every CAction a which returns a representation $r_a \in \mathscr{R}$ of a with respect to a given preference.

The set of important CAction-preference-pairs with respect to a given context c is defined to be the union of the result of all CLocal context services:

$$ap(c) := \bigcup_{\substack{s \in \mathscr{S}_{\mathscr{C}},\text{ and} \\ s \text{ is CLocal with respect to } c}} s(c)$$

The set $ap(c)$ contains all CAction-preference-pairs of CLocal context services, that is of those services which are assumed to be of importance for the current situation. Afterwards, the set $ap(c)$ is sorted with respect to the preferences attached to its elements, resulting in the list $ap'(c)$:

$$ap'(c) := \big[(a,p) \in ap(c),\ \text{sorted with respect to } p\big]$$

The overall result of the CASA-system with respect to a given context c is the following compound representation:

$$\begin{aligned} CASA : \mathscr{C} &\to \mathscr{R} \\ c &\mapsto \circ_{(a,p) \in ap'(c)} \mathrm{rep}_a(p) \end{aligned}$$

Every possible CAction-preference-pair is computed (ap) and the resulting list sorted with respect to the preferences (ap'). For each CAction a, an appropriate representation is constructed depending on the attached preference p, that is we compute $\mathrm{rep}_a(p)$ for every CAction-preference pair (a,p). Those representations are composed using the composition function $\circ$, resulting in the final representation to be presented to the user.

3.5. CASA design resume

While revisiting our challenge problems from the introduction, we discuss the overall design of the system from a more informed perspective and relate the design decisions to our observations.

C1. By using web-based frontends for services, every mobile Internet-enabled device can interact with these services (see observation *O1*).

C2. By utilising the CASA-approach, important services are discovered and selected based on the current situation in general (see observation *O2*).

C3. By employing user contributed context services, all available information and services are collected and filtered with respect to the current context (see observation *O3*).

C4. By compiling a homogeneous representation from the results of the services, the user does not perceive the devices but their provided assistance (see observation *O4*).

C5. By designing the system as open and community-driven, even heterogeneous large-scale environments of devices and services can be maintained (see observation *O5*).

The CASA-approach as described above is based on a context store holding the current user context in a format suitable for inference. The context is updated by using sensors attached to locations. Based on this information, all locally available context services (CServices) are evaluated to compute a set of important context actions (CActions). After ordering the actions, a suitable representation is aggregated and shown to the user.

4. Use Cases

Below two realised use cases are presented and discussed, in which the CASA approach has been applied. At first, additional situation dependent functionality is added into a learning management system. Then it is shown, how a controller of smart meeting rooms can be understood as CASA-instance.

4.1. CASA within learning management systems

A prototype of the CASA-approach has been implemented based on the learning management software Stud.IP.[2] This open source system is used at over 50 universities to manage resources and to provide additional material to students. In total there are over 450.000 registered users.[3] Figure 2 contains a screenshot of our implemented system, showing parts of the standard Stud.IP web site enhanced by four services for device interaction provided by the CASA system (shown in the lower part with icons on the left). Please note, that the usage of additional services, like control of presentation equipment, is not part of the Stud.IP system, but has been integrated by the CASA system.

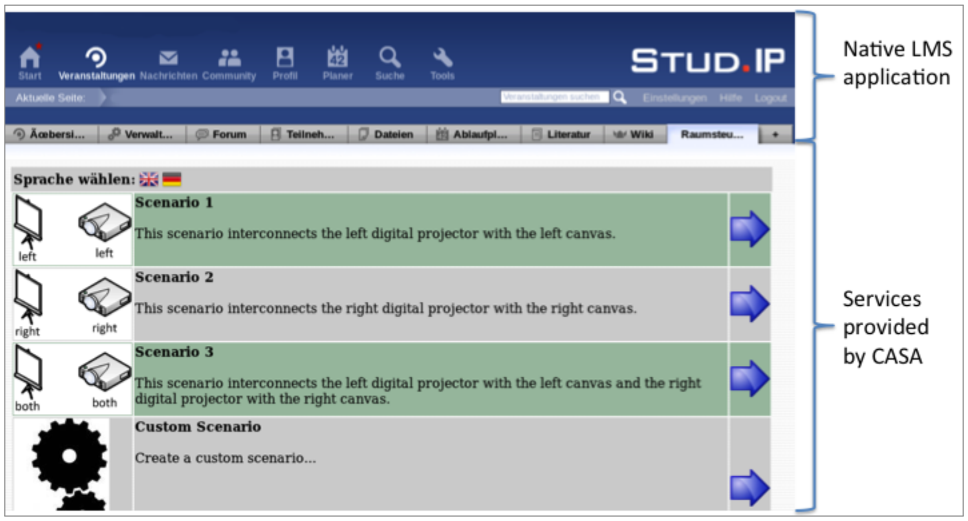
The necessary position sensor here is integrated in the Stud.IP as it senses the login of a teacher which then is processed by the system in connection with the next lecture and the room which is reserved. As shown in figure 3, this was done by implementing export services into Stud.IP, which were called by importer modules of the CASA node. This room has multiple digital projectors and devices. To access them plug-ins were developed. These were connected to an aggregation module at the CASA node and a web site plug-in on the Stud.IP site to create the shown web interface for the available services.

The implementation can integrate service interfaces like device controls or media players. These relate directly to the devices of the room or to the situation context if there is a lecture streamed at the same time. The interfaces are the ones which are standard for the service, like the media player, or they are created by adapters which realize the interface to the devices. This encapsulates their functionality as CActions and therefore allows a uniform representation of interactions as shown in the screenshot where the different projector configurations can be chosen. While these services are provided reactively by opening the lecture sites in a specified situation, there are also notifications which can be created by the system proactively to provide a specific information to the user if a specified situation occurs.

This is a complex scenario as it was necessary to implement the adapters, but we also realized an interface for the users of the system to integrate simple services like videos, document readers and web sites. The users here classify the situations with location, lecture and user role.

The CASA context store and evaluation system is implemented using the business rules management system JBoss Drools[4] as rules engine. Drools is well documented and has an active community. Thus

[2]`http://www.studip.de/`
[3]`http://www.studip.de/presse`
[4]`http://www.jboss.org/drools`



Sprache wählen:

Scenario 1

This scenario interconnects the left digital projector with the left canvas.

Scenario 2

This scenario interconnects the right digital projector with the right canvas.

Scenario 3

This scenario interconnects the left digital projector with the left canvas and the right digital projector with the right canvas.

Custom Scenario

Create a custom scenario...

Fig. 2. Screenshot with device controls integrated into the Stud.IP web site.

we were able to combine it easily with Stud.IP and our context sources. The existing concepts of Stud.IP are mapped to a generic object-oriented ontology and then sent to the Drools database. This allows the interconnection of knowledge from different sources which are integrated into one world model, hosted by the rules engine. The facts which are collected by the sensors are evaluated by rules which specify the relations and transformations to adapt the fact to our world model.

The CServices (rules) as well as the context store are implemented in Drools using if-then-rules. They relate to the facts which are stored by the system. The code snippet in figure 4 shows a rule which informs a user of the lactose free menues in the cafeteria. Beside the proactive user notification the system also provides interfaces for other applications to use the stored context. The code in figure 5 provides the next course for a given user and time.

We use the web site of Stud.IP for aggregation of the relevant services. As it depends on the users' role the security aspects (authentication, authorization and accounting) are already realized. While building the Stud.IP page, the CASA system is queried with the current context (lecture and user name). The result is a HTML snippet to be embedded into the page. Multiple results are ordered by their priorities.

4.2. Goalaviour-based control as CASA-instance

The control of smart environments is a challenging task. Numerous approaches have been presented and for a general introduction and review of the state of the art, we refer to (Cook & Das, 2005; Nakashima, Aghajan, & Augusto, 2010; Poslad, 2009). In (Bader and Dyrba (2011)) a novel approach has been presented. Inspired by Brooks subsumption architecture (Brooks, 1986), the *goalaviour*-based control system for smart environments has been developed. In particular, it has been applied to control the smart appliance laboratory shown in figure 6. The room features numerous sensors and actuators, among them are, for example, an indoor positioning system, moveable screens and multiple projectors. The scheme of the overall system is depicted in figure 7.

Instead of controlling the actuators of the environment directly (as in Brooks approach) the behaviours produce goals. Therefore, they are called *goalaviours* – *goal*-producing beh*aviours*. Every goalaviour subscribes within our middleware (Bader & Kirste, 2013) to important sensors deployed into the environment. Thus, it is able to maintain a representation of the important subset of the state of the world. As soon as the sensor inputs imply a new goal, this goal is emitted by the goalaviour. A central controller collects all

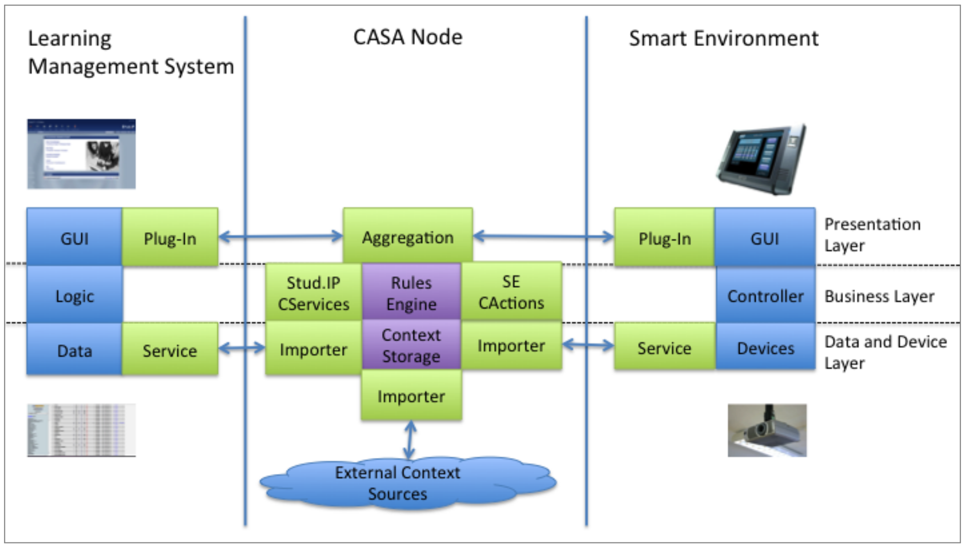


Fig. 3. Architecture of the Stud.IP use case.

```
rule "LactoseFreeMenuNotification"
  when
    c : Cafeteria(l_free_menu_ava. == true)
    m : MobilePhone( location == c )
    p : Person( activeDevice == m, allergy == "lactose" )
  then
    m.notification(c.lactosefree_menu);
end
```

Fig. 4. Drools code to trigger a user specific menu notification.

goals from the attached goalaviours. Those goals are merged resulting in a set of final goals, which is given to a planner. Merging in necessary as described in (Bader & Dyrba, 2011) to ensure the consistency of the controller. Then, the resulting plan is executed to actually achieve the goals by influencing the world. A user study performed to evaluate the controller (Dyrba, Nicolay, Bader, & Kirste, 2011) showed the general acceptance of automatic control systems.

Originally, the control system has been designed to perform all necessary actions directly. That is, as soon as a goalaviour emits a goal, the controller tries to achieve it. As an alternative, the control system can be understood and be set up as a CASA-instance as follows:

1. Every goalaviour g maintains its own context representation $\mathscr{C}_g$. This functionality is part of the middleware, which allows state tracking of components by means of device-property-value triples holding the current value of all subscribed device properties as described in (Bader & Nyolt, 2012).
2. Every goalaviour emits a set of goals if necessary.
3. A central controller collects all goals and merges them to produce an overall goal to be achieved by the environment.
4. Every goal within the final set of goals is transformed into a CAction. Those CActions, allow the user to invoke the underlying goal. The goal is achieved by executing an appropriate sequence of device actions.
5. A final representation is constructed as shown in figure 8. The resulting GUI is an aggregation of all currently open goals.

```
query "GetNextCourseByUsername"
            (String username, Date d)
  p : Person (p.username == username)
  l : Lecture (begin >= d)
  not Lecture (begin >= d, l.begin > begin)
  c : Course (c.members contains p, c.lectures contains l)
end
```

Fig. 5. Drools query code to access the next lecture for a given user and date.

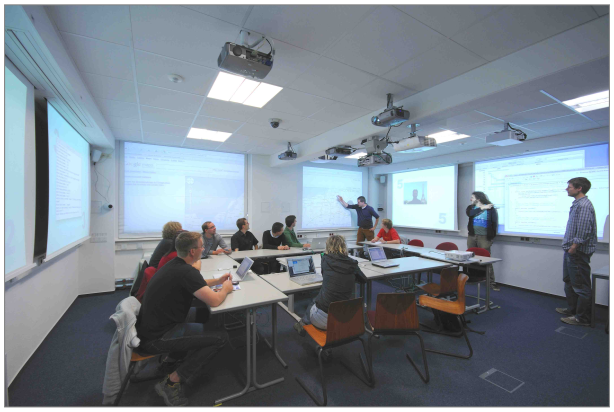


Fig. 6. The Smart Appliance Lab a prototype of a future meeting room.

All currently open goals are collected, sorted according to their preference and are presented to the user. The system replaces the normal interaction by a goal-oriented approach. Usually a sequence of device actions has to be performed to obtain a given goal. Here, only the goals themselves are presented to the user and an invocation of the corresponding CAction results in the execution of the underlying sequence of device actions. This enables an intuitive control of the environment, because the user does not have to know the details of the environment, and does not have to know how to achieve its goals.

Goalaviours itself can be implemented in various ways. Figure 10 shows the internal structure of a goalaviour implemented as a simple state machine. To actually create and deploy such a goalaviour into the environment, the commands shown in figure 9 have to be invoked within our deployment system. The deployment system itself is web based which allows to create new goalaviours by entering the shown commands into a web form.

For all further details concerning the goalaviour based control, we refer to (Bader & Dyrba, 2011) and summarize here the relation to the CASA-approach introduced above by recapitulating the challenge problems from the introduction:

C1. By using web-based frontends to actually invoke the goals, every mobile Internet-enabled device can easily interact with the controller.

C2. Currently important goals (thus underlying services) are discovered and selected based on the current situation, which depends on the current user locations and state of embedded devices.

C3. Goalaviours can be deployed into the system dynamically, thus allowing users to contribute own CServices.

C4. By compiling a homogeneous representation, the user does not perceive the devices but their provided assistance by means of open goals.

C5. We believe that the goalaviour based approach can also be applied to larger environments. Goalaviours

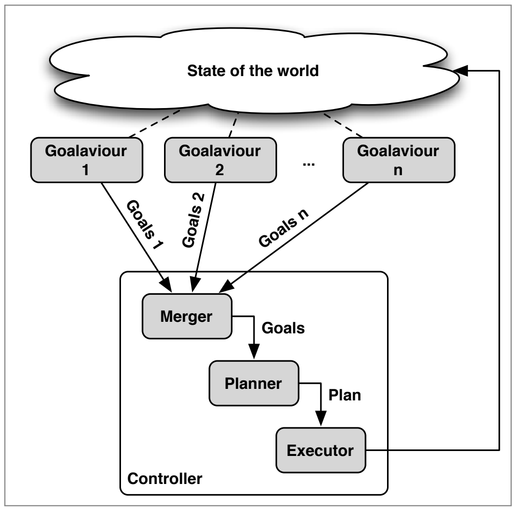


Fig. 7. General scheme of a goalaviour-based controller.

are deployed into the environment as stand alone processes and subscribe themselves to all important events via a publish/subscribe system (Bader & Nyolt, 2012).

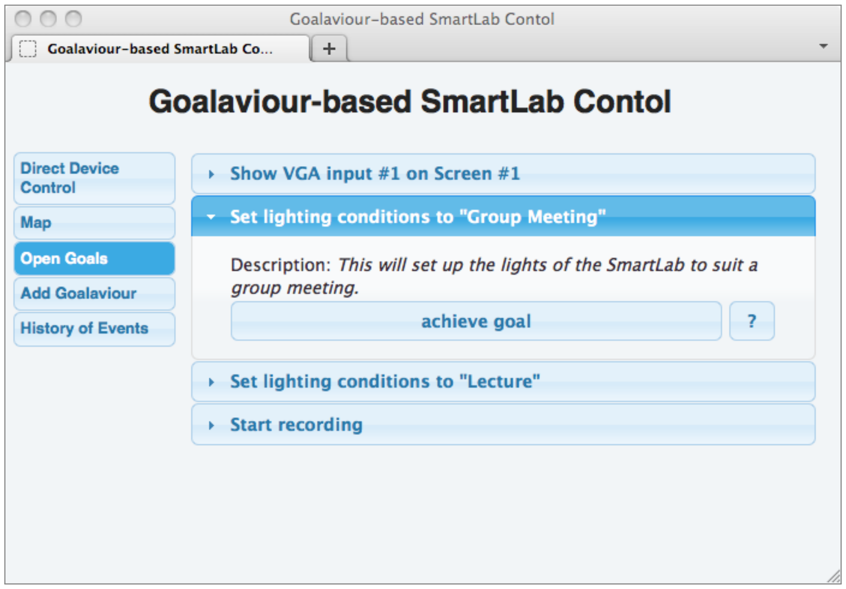
5. Conclusion and Future Work

The CASA system, as described above, solves the challenge problems as stated in the introduction. Mobile users can employ and interact with ubiquitous services by means of a context dependent aggregation of CActions (C1). Those allow the user to interact with services provided by the environment using a common representation. By using the aggregated web site of the CASA system, the user is able to discover (probably) important services provided at and for the current location (C2). Using the context-based filtering and sorting of the system, it is easy to identify the relevant services, because they are shown on a prominent position (C3). By utilizing CActions notifying the user, users can be directed to services in their proximity without even being aware of them.

We believe that crowdsourcing is the best answer to the management and maintenance of ecosystems of heterogeneous devices and services. Therefore, the CASA system relies heavily on standard web technologies, not only to build interfaces for the users, but also to add services to the environment.

As illustration of the added value, which is provided by the system, we discussed two realized use cases. The first illustrates the applicability of our approach to existing smart environments and systems with large amounts of contextual information. The combination of both via the CASA approach allows an enrichment of the functionality while reducing the configuration effort and facilitating the usage. In addition to this the CASA node allows a further extension of the compound system. This was also a motivating fact for the realization of the second use case. In addition to the encapsulation of existing functionalities as services, the CASA approach introduced the method of crowdsourcing for generating new goalaviours. The aggregation of (possibly) relevant goalaviours with relation to the context enhances the assistance of the system. It allows beside the execution of the proved wanted assistance also the offering of actions which are probably wanted. We believe that the provided approach is also applicable to other cloud services as they have also web-based interfaces which need to be accessed at user-specified situations.

Nonetheless, a number of problems are still not solved and need to be examined in the future. The underlying ontology and the rules which are realized might need to be replaced by other storage and inferences systems. Potential candidates are description logic based ontologies together with suitable reasoning


Goalaviour-based SmartLab Contol
Goalaviour-based SmartLab Co...
Goalaviour-based SmartLab Contol
Direct Device Control
Map
Open Goals
Add Goalaviour
History of Events
Show VGA input #1 on Screen #1
Set lighting conditions to "Group Meeting"
achieve goal
?
Set lighting conditions to "Lecture"
Start recording

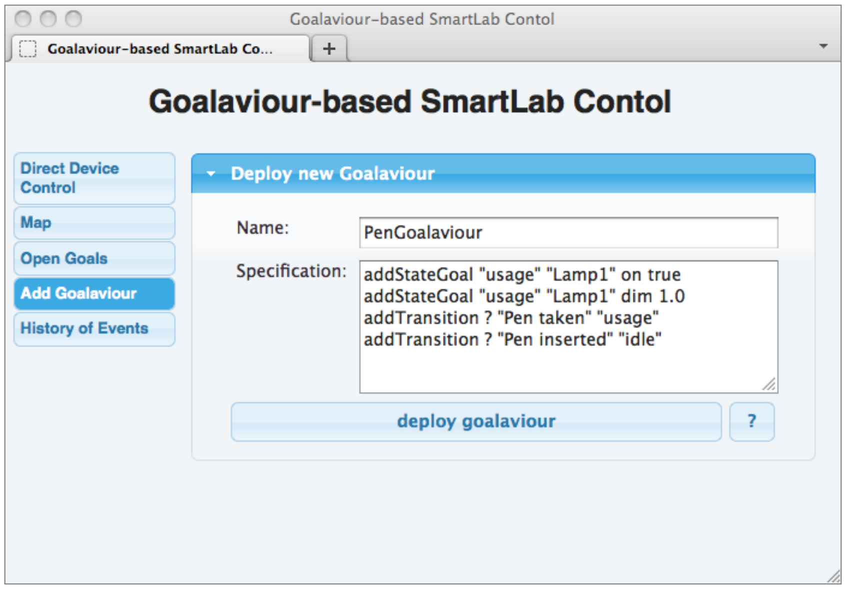
Description: *This will set up the lights of the SmartLab to suit a group meeting.*

Fig. 8. Screen-shot of the prototypic CASA-GUI attached to a goalaviour-based control system. In addition to a direct control of the devices, the GUI offers a goal-based interaction. All open goals are collected and presented to the user. The GUI itself is implemented as a HTML5 webpage to be accessible by as many devices as possible.

systems. Here we need some further work to evaluate the applicability of other approaches like for instance OWL. To accommodate noisy sensor inputs, probabilistic reasoning might be necessary. This will also lead to an extended review of the weighting system, which is currently quite simple as it only uses the weighting of the rules as order. Here more fine-grained approaches or the consideration of heuristics could further simplify the user experience by highlighting the most important information. Observing the user behavior might also be used to gather service sets and relevant situations. Machine learning methods could also be an interesting approach to generate basic context and rules. The development of adapters is also important as an automatic generation for services would save time but lacks in the usability and does not hide the complexity of the offered service. Additionally, there are still challenging problems in the generation of service representations, which rely on non web-based service technologies like Bluetooth or ZigBee (Zender, Lucke, & Tavangarian, 2010). The situation-aware integration of cloud-based resources like storage is also a challenge as it needs more platform-specific implementations on the sensor and integration side.

Beside the user interaction there is also the machine interaction with the system as context provider. Here we need to find an approach which supports the usage of the context while the ontology is changed by the crowd. At last there is the need for a web application which acts as the interface to the crowd as well as beginning point for extensive tests and evaluations. Here we plan to combine a wiki, an OpenStreetMap system, and an android application. With the wiki as interface for generating CServices and CActions, the android application as user interface, and the OpenStreetMap system as additional context provider, we hope to attract a suitable amount of users to test and to evaluate our approach on a larger scale.

In this work, we presented our CASA approach for a context-aware service access. After describing the general idea behind CASA, we discussed two use cases: a novel approach to use services context-aware inside a learning management system and as an interface for a goalaviour-based smart environment. The CASA approach therefore can be seen as a system to interweave different smart environment services and contexts. We discussed different challenges, which arise in large-scale pervasive environments and advocated for a crowd-sourced development. This allows a scalable system which can be adapted to new situations, context sensors, services and locations. As concept building blocks we presented the different



Name: PenGoalaviour

Specification:

```
addStateGoal "usage" "Lamp1" on true
addStateGoal "usage" "Lamp1" dim 1.0
addTransition ? "Pen taken" "usage"
addTransition ? "Pen inserted" "idle"
```

Fig. 9. Screen-shot of the GUI to create and deploy the whiteboard goalaviour shown in Figure 10.

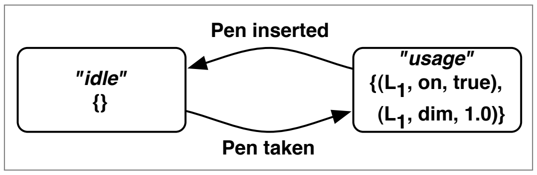


Fig. 10. A simple two state goalaviour to control a lamp close to some whiteboard. The whiteboard should be illuminated whenever the pen is taken. After reinserting the pen into its holder, no goals are produced. The goals are shown below the state labels.

parts of the CASA system as well as examples to illustrate their usage and relevance. Finally, we discussed some future works.

References

Atzori, L., Iera, A., & Morabito, G. (2010, October). The internet of things: A survey. *Comput. Netw.*, *54*, 2787–2805. Retrieved from http://dx.doi.org/10.1016/j.comnet.2010.05.010 doi: 10.1016/j.comnet.2010.05.010

Bader, S., & Dyrba, M. (2011, July). Goalaviour-based control of heterogeneous and distributed smart environments. In *Proceedings of the 7th international conference on intelligent environments - ie'11* (pp. 142–148). IEEE Computer Society. doi: 10.1109/IE.2011.33

Bader, S., & Kirste, T. (2013, June). An overview of the HELFERLEIN-system. In *Proceedings of MMS 2013.* Berlin, Germany.

Bader, S., & Nyolt, M. (2012). A context-aware publish-subscribe middleware for distributed smart environments. In *Proceedings of MUCS2012.* Retrieved from http://ubiquitous-management.org/mucs/2012/index.php

Baldauf, M., Dustdar, S., & Rosenberg, F. (2007). A survey on context-aware systems. *Int. Journal of Ad Hoc and Ubiquitous Computing*, *2*(4), 263–277.

Bellavista, P., Corradi, A., Fanelli, M., & Foschini, L. (2013). A Survey of Context Data Distribution for Mobile Ubiquitous Systems. *ACM Computing Surveys*, *45*(1). Retrieved from http://www-lia.deis.unibo.it/Staff/LucaFoschini/pdfDocs/context_survey_CSUR.pdf

Berchtold, M., Budde, M., Gordon, D., Schmidtke, H., & Beigl, M. (2010). Actiserv: Activity recognition service for mobile phones. In *Wearable computers (ISWC), international symposium on* (pp. 1–8).

Brezmes, T., Gorricho, J.-L., & Cotrina, J. (2009). Activity recognition from accelerometer data on a mobile phone. In *Distributed computing, artificial intelligence, bioinformatics, soft computing, and ambient assisted living* (Vol. 5518, p. 796-799). Springer Berlin / Heidelberg.

Brooks, R. A. (1986). A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, *RA-2*(1), 14–23. Retrieved from http://people.csail.mit.edu/brooks/papers/AIM-864.pdf

Chon, J., & Cha, H. (2011). Lifemap: A smartphone-based context provider for location-based services. *IEEE Pervasive Computing*, *10*(2), 58–67.

Cook, D. J., & Das, S. K. (2005). *Smart environments*. Wiley.

Dey, A., & Abowd, G. (2000). Towards a better understanding of context and context-awareness. In *CHI 2000 workshop on the what, who, where, when, and how of context-awareness* (Vol. 4, pp. 1–6).

Dyrba, M., Nicolay, R., Bader, S., & Kirste, T. (2011, December). Evaluation of two control systems for smart environments. In *Proceedings of the 8th international icst conference on mobile and ubiquitous systems: Computing, networking and services - 3rd workshop on context systems design, evaluation and optimisation.* Copenhagen, Denmark.

Henpraserttae, A., Thiemjarus, S., & Marukatat, S. (2011, may). Accurate activity recognition using a mobile phone regardless of device orientation and location. In *Body sensor networks (BSN), 2011 international conference on* (p. 41 -46).

Kiani, S. L., Moltchanov, B., Knappmeyer, M., & Baker, N. (2011, February). Large–scale context-aware system in smart spaces: Issues and challenges. In *Proceedings of the baltic congress on future internet and communications (BCFIC 2011) – special session on smart spaces and ubiquitous solutions.* Riga, Latvia.

Kjær, K. (2007). A survey of context-aware middleware. In *Proceedings of the 25th conference on iasted international multi-conference: Software engineering* (pp. 148–155).

Knappmeyer, M., Kiani, S. L., Reetz, E. S., Baker, N., & Tonjes, R. (2013). Survey of Context Provisioning Middleware. *IEEE Communications Surveys & Tutorials*, *15*(3), 1492–1519. Retrieved from http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6422290 doi: 10.1109/SURV.2013.010413.00207

Nakashima, H., Aghajan, H., & Augusto, J. C. (Eds.). (2010). *Handbook of ambient intelligence and smart environments*. Springer.

Patterson, D. J., Liao, L., Gajos, K., Collier, M., Livic, N., Olson, K., … Kautz, H. (2004). Opportunity knocks: a system to provide cognitive assistance with transportation services. In *In international conference on ubiquitous computing (UbiComp)* (pp. 433–450). Springer.

Poslad, S. (2009). *Ubiquitous computing: Smart devices, environments and interactions*. Wiley.

Strobbe, M., Laere, O. V., Ongenae, F., Dauwe, S., Dhoedt, B., Turck, F. D., … Luyten, K. (2011). Integrating location and context information for novel personalised applications. *IEEE Perv. Comp.*, *99*(PrePrints). doi: 10.1109/MPRV.2011.60

Zender, R., Lucke, U., & Tavangarian, D. (2010). SOA interoperability for large-scale pervasive environments. *Advanced Information Networking and Applications Workshops, International Conference on*, *0*, 545-550. doi: 10.1109/WAINA.2010.128

Author Biographies

Philipp Lehsten is a Ph.D. student at the Research Group Computer Architecture at the University of Rostock and member of the Research Training Group Multimodal Smart Appliance Ensembles for Mobile Applications" (MuSAMA), which is funded by the German National Research Foundation (DFG). He received his M.Sc. in computer engineering at the University of Rostock (Germany) in 2009. His research interests include service-oriented architectures (SOA), context-aware systems, and the user-driven dissemination and integration of services in smart environments.

Sebastian Bader is a post-doctoral researcher in the graduate school MuSAMA and the chair of Mobile Multimedia Information Systems at the University of Rostock. He works on smart environments in which assistance emerges from a dynamic ad-hoc ensemble of co-located heterogeneous devices. He completed his Ph.D. at the Technische Universitt Dresden in 2009. In addition to his research on smart environments, he works in the areas of neural-symbolic integration, middleware systems, and probabilistic inference. Sebastian served in committees for several international conferences, journals and workshop in the areas of smart environments and artificial intelligence. He authored about 50 peer-reviewed articles.

Dr. Djamshid Tavangarian is a Professor at the University of Rostock (Germany) where he represents the research area of Computer Architecture. The current main topics of his research activities concentrate on computer architectures for distributed systems, pervasive computing, adaptive and embedded systems, wireless communication systems and also, eLearning and multimedia architectures for mobile learning. Dr. Tavangarian is author and co-author of more than 300 scientific publications. He is member of several scientific organizations..